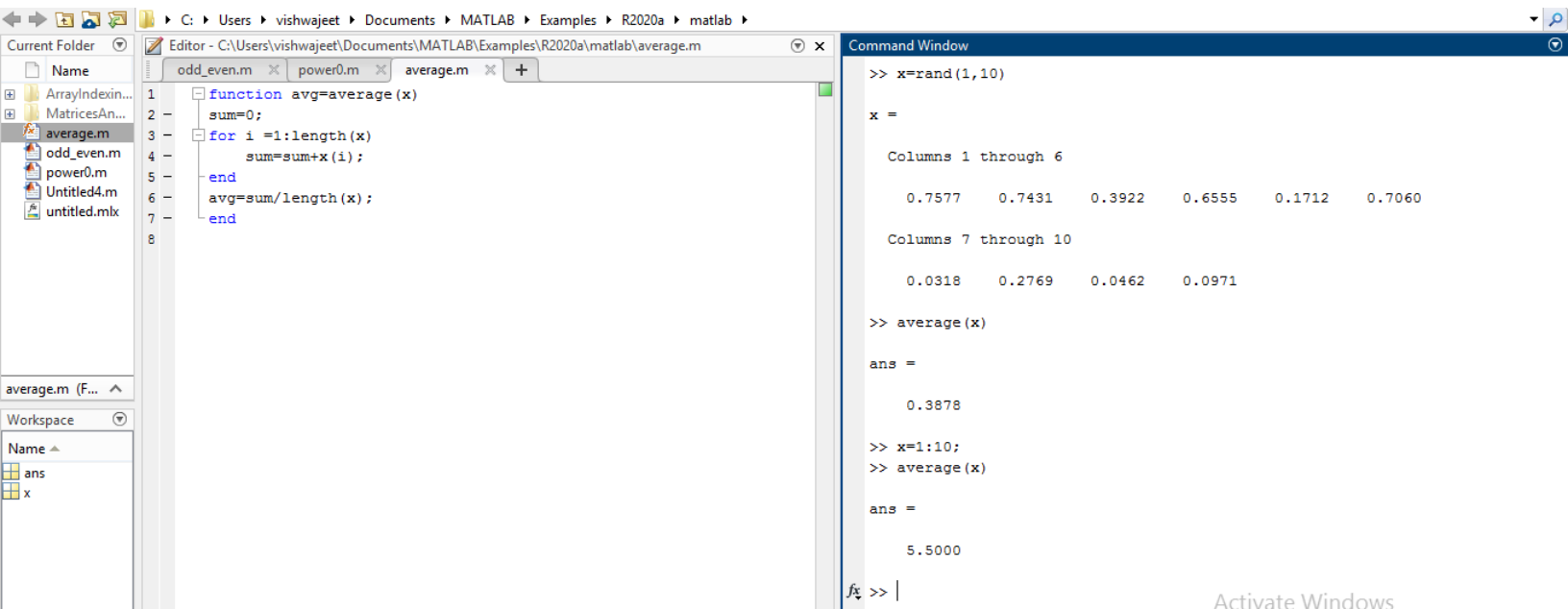


# Abhijeet Deshmukh

mis:111909002

## Q\_1:average of 10 no.



The image shows a MATLAB IDE window with the following components:

- Editor:** Displays a function file named `average.m` with the following code:

```
1 function avg=average(x)
2     sum=0;
3     for i =1:length(x)
4         sum=sum+x(i);
5     end
6     avg=sum/length(x);
7 end
8
```
- Command Window:** Shows the execution of the function with two test cases:

```
>> x=rand(1,10)
x =
Columns 1 through 6
    0.7577    0.7431    0.3922    0.6555    0.1712    0.7060
Columns 7 through 10
    0.0318    0.2769    0.0462    0.0971

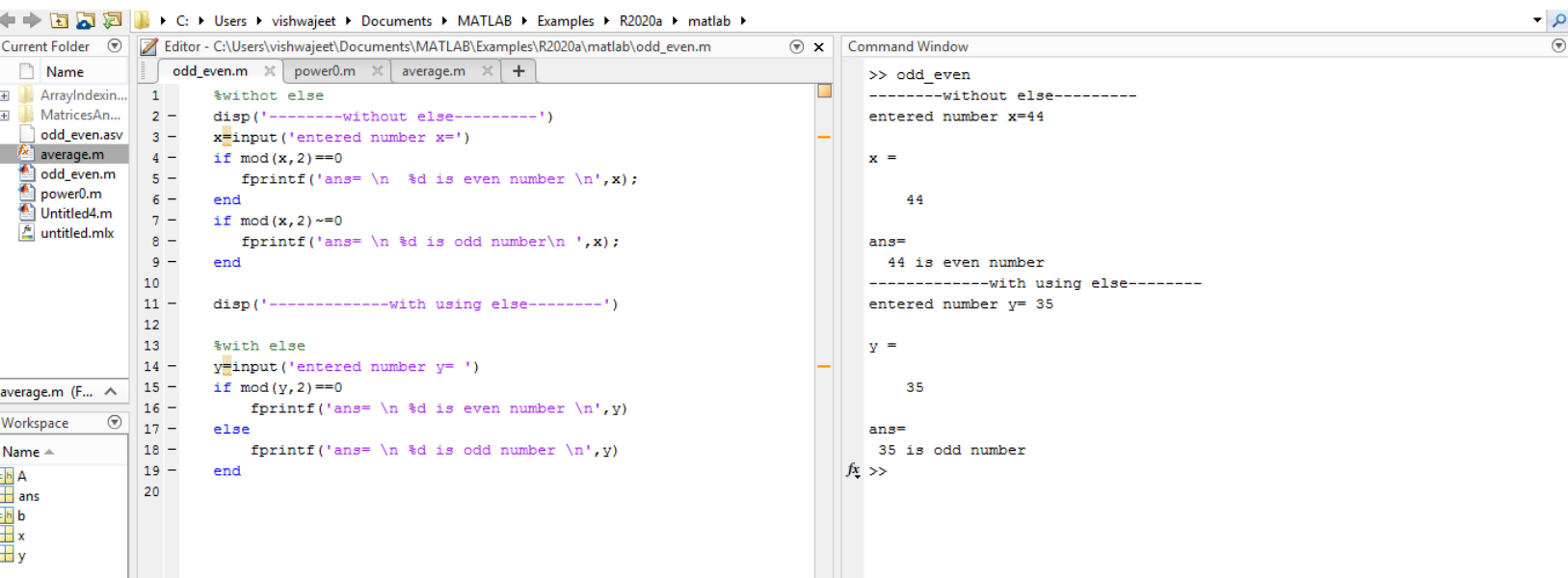
>> average(x)
ans =
    0.3878

>> x=1:10;
>> average(x)
ans =
    5.5000

fx >> |
```
- Workspace:** Lists the variables `ans` and `x`.
- Current Folder:** Lists the files `average.m`, `odd_even.m`, `power0.m`, `Untitled4.m`, and `untitled.mlx`.

Activate Windows

# Q2: Odd\_Even



The image shows a MATLAB environment with the Editor and Command Window. The Editor displays a script named `odd_even.m` with the following code:

```
1 %without else
2 disp('-----without else-----')
3 x=input('entered number x=')
4 if mod(x,2)==0
5     fprintf('ans= \n %d is even number \n',x);
6 end
7 if mod(x,2)~=0
8     fprintf('ans= \n %d is odd number\n ',x);
9 end
10
11 disp('-----with using else-----')
12
13 %with else
14 y=input('entered number y= ')
15 if mod(y,2)==0
16     fprintf('ans= \n %d is even number \n',y)
17 else
18     fprintf('ans= \n %d is odd number \n',y)
19 end
20
```

The Command Window shows the execution of the script:

```
>> odd_even
-----without else-----
entered number x=44

x =

    44

ans=
    44 is even number
-----with using else-----
entered number y= 35

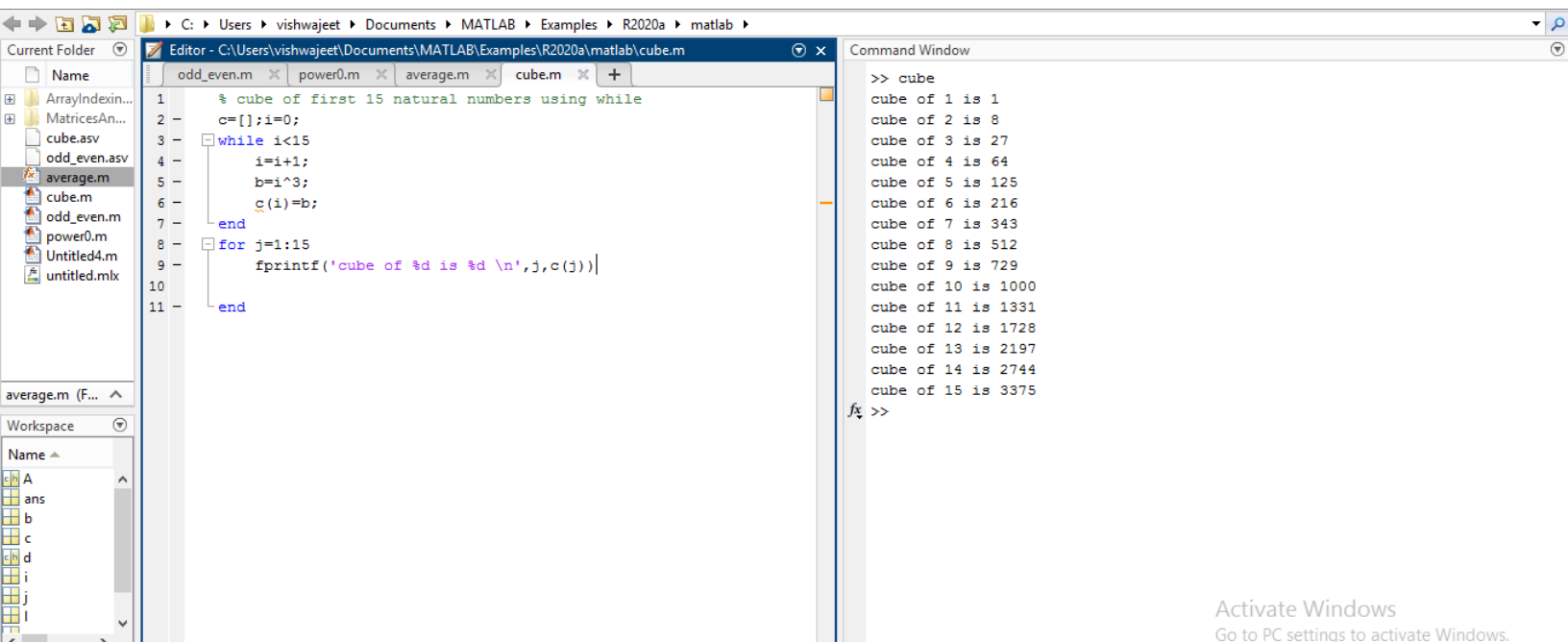
y =

    35

ans=
    35 is odd number
fx >>
```

The Workspace window on the left shows the following variables: `A`, `ans`, `b`, `x`, and `y`.

# Q3:cube upto 15



The image shows a MATLAB environment with the Editor and Command Window. The Editor displays a script named 'cube.m' with the following code:

```
1 % cube of first 15 natural numbers using while
2 c=[];i=0;
3 while i<15
4     i=i+1;
5     b=i^3;
6     c(i)=b;
7 end
8 for j=1:15
9     fprintf('cube of %d is %d \n',j,c(j))
10 end
11
```

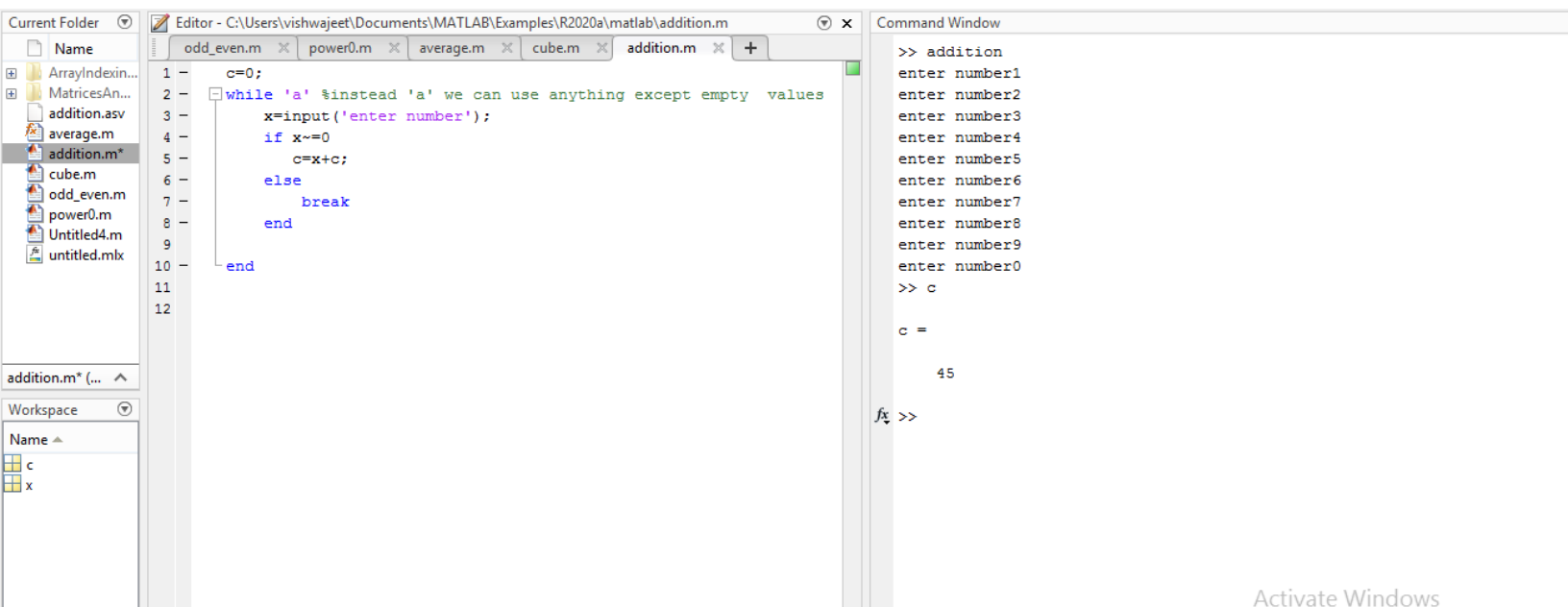
The Command Window shows the output of the script:

```
>> cube
cube of 1 is 1
cube of 2 is 8
cube of 3 is 27
cube of 4 is 64
cube of 5 is 125
cube of 6 is 216
cube of 7 is 343
cube of 8 is 512
cube of 9 is 729
cube of 10 is 1000
cube of 11 is 1331
cube of 12 is 1728
cube of 13 is 2197
cube of 14 is 2744
cube of 15 is 3375
fx >>
```

The Workspace panel on the left shows the following variables: A, ans, b, c, d, i, j, l.

Activate Windows  
Go to PC settings to activate Windows.

# Q4: adding numbers untill zero



The image shows the MATLAB environment with the Editor and Command Window. The Editor displays a script named `addition.m` with the following code:

```
1 c=0;
2 while 'a' %instead 'a' we can use anything except empty values
3     x=input('enter number');
4     if x~=0
5         c=x+c;
6     else
7         break
8     end
9
10 end
```

The Command Window shows the execution of the script:

```
>> addition
enter number1
enter number2
enter number3
enter number4
enter number5
enter number6
enter number7
enter number8
enter number9
enter number0
>> c

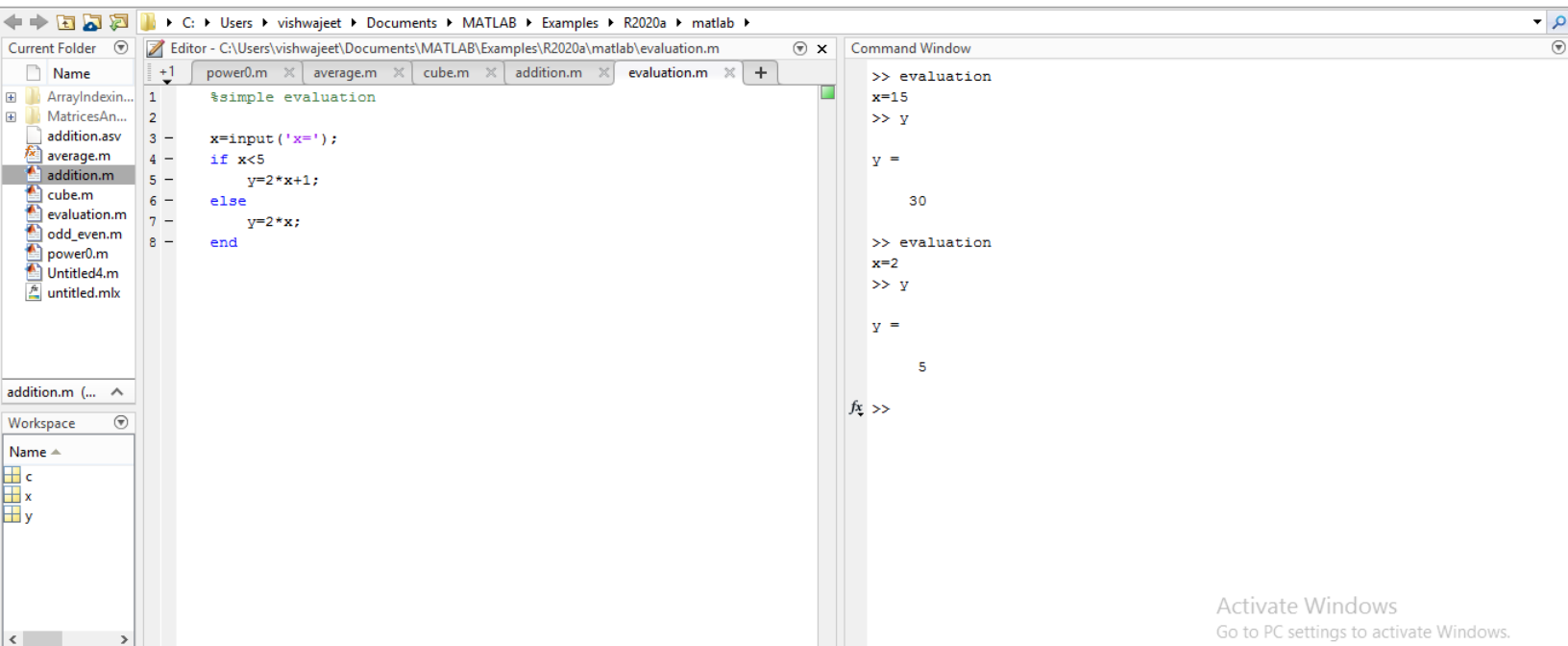
c =

    45
```

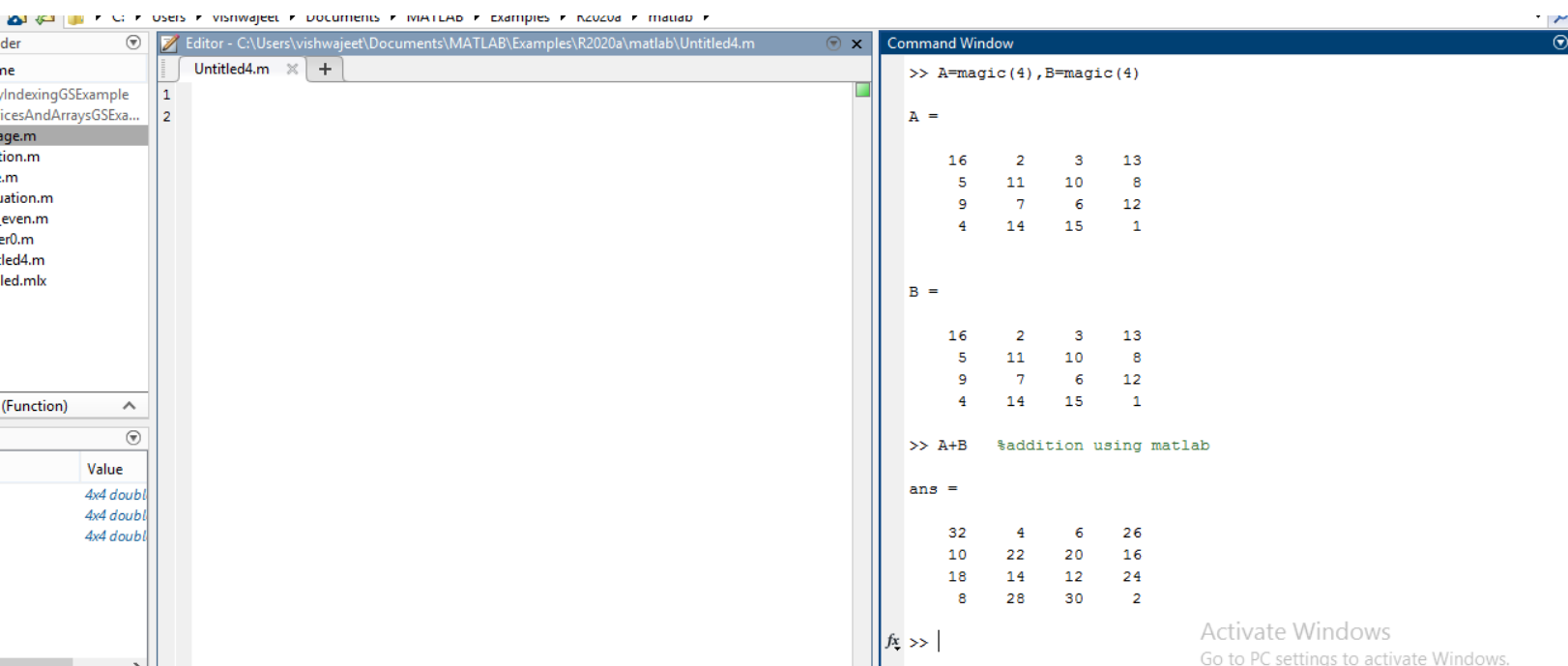
The Workspace window shows the variables `c` and `x`.

Activate Windows

# Q5: evaluation



# Q6-->(i) using MATLAB as tool



The image shows the MATLAB R2020a interface. The Editor window displays a script named 'Untitled4.m' with two lines of code. The Command Window shows the execution of these commands, resulting in the creation of two 4x4 magic square matrices, A and B, and their sum, ans.

```
Editor - C:\Users\vishwajeet\Documents\MATLAB\Examples\R2020a\matlab\Untitled4.m
Untitled4.m
1
2

Command Window
>> A=magic(4),B=magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

B =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> A+B %addition using matlab

ans =

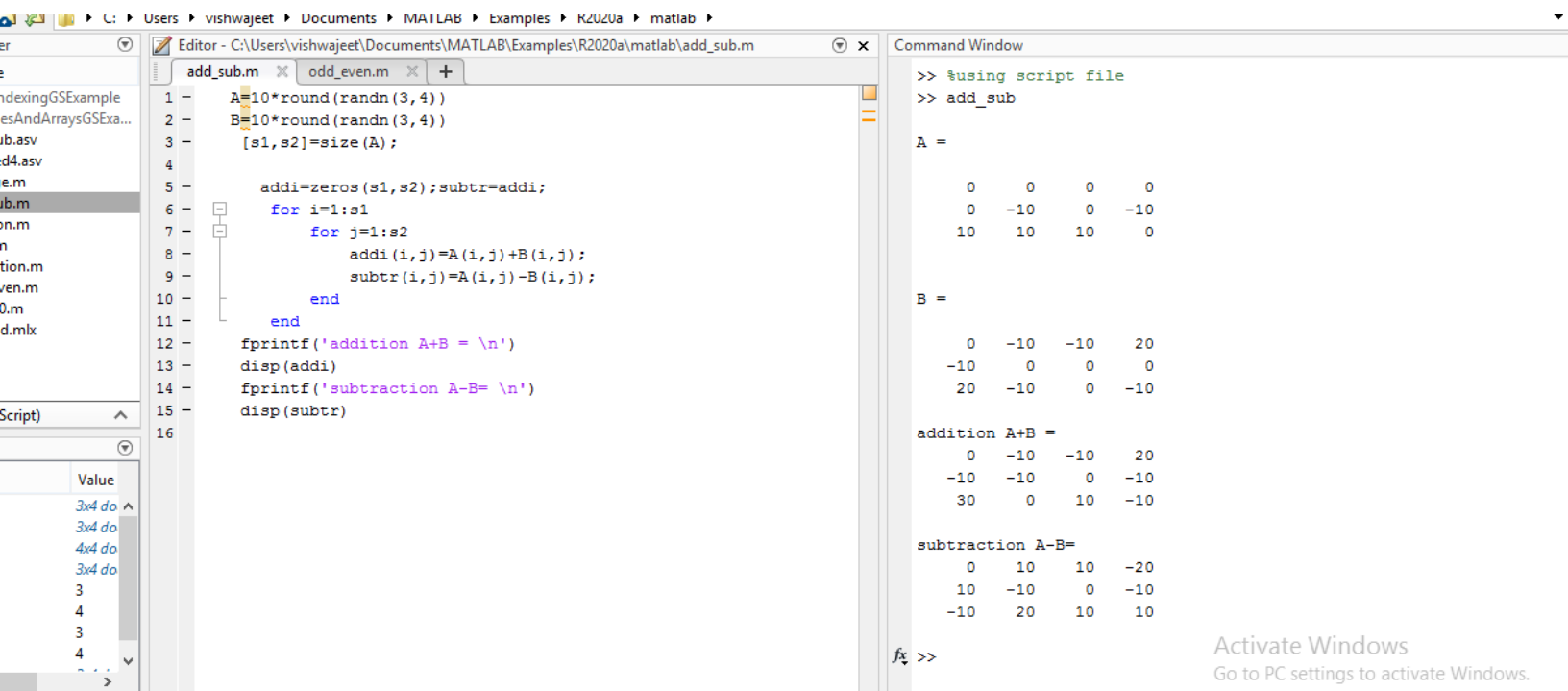
    32     4     6    26
    10    22    20    16
    18    14    12    24
     8    28    30     2

fx >> |
```

Value  
4x4 double  
4x4 double  
4x4 double

Activate Windows  
Go to PC settings to activate Windows.

# Q6-->(ii) add,sub using script file



The image shows the MATLAB environment with the Editor and Command Window. The Editor displays a script file named `add_sub.m` with the following code:

```
1 A=10*round(randn(3,4))
2 B=10*round(randn(3,4))
3 [s1,s2]=size(A);
4
5 addi=zeros(s1,s2);subtr=addi;
6 for i=1:s1
7     for j=1:s2
8         addi(i,j)=A(i,j)+B(i,j);
9         subtr(i,j)=A(i,j)-B(i,j);
10    end
11 end
12 fprintf('addition A+B = \n')
13 disp(addi)
14 fprintf('subtraction A-B= \n')
15 disp(subtr)
16
```

The Command Window shows the execution of the script:

```
>> %using script file
>> add_sub

A =

     0     0     0     0
     0    -10     0    -10
    10    10    10     0

B =

     0    -10    -10    20
    -10     0     0     0
    20    -10     0    -10

addition A+B =

     0    -10    -10    20
    -10    -10     0    -10
    30     0    10    -10

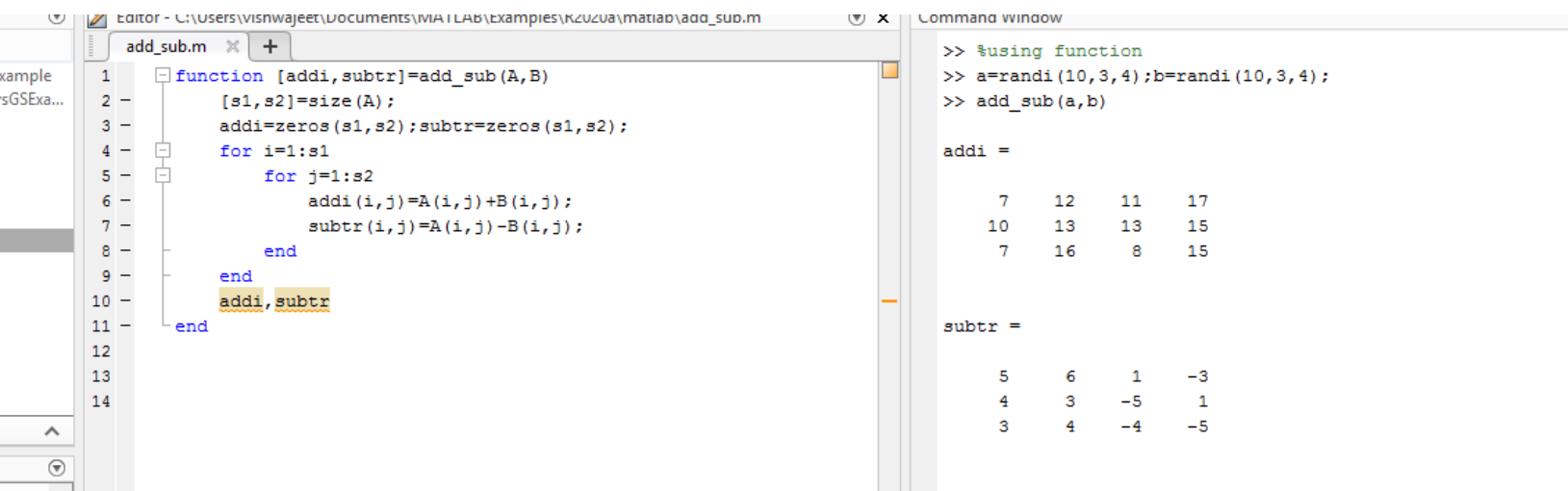
subtraction A-B=

     0    10    10    -20
    10    -10     0    -10
    -10    20    10    10

fx >>
```

An "Activate Windows" watermark is visible in the bottom right corner of the Command Window.

# Q6--(iii) add,sub using function



The image shows a MATLAB environment with an Editor window and a Command Window. The Editor window displays a function named `add_sub` that takes two matrices `A` and `B` as inputs and returns their sum `addi` and difference `subtr`. The function uses nested loops to calculate the sum and difference element-wise. The Command Window shows the execution of the function with random matrices `a` and `b`, displaying the resulting `addi` and `subtr` matrices.

```
Editor - C:\Users\visnwajeet\Documents\MATLAB\examples\K2020a\matlab\add_sub.m
add_sub.m
1 function [addi,subtr]=add_sub(A,B)
2     [s1,s2]=size(A);
3     addi=zeros(s1,s2);subtr=zeros(s1,s2);
4     for i=1:s1
5         for j=1:s2
6             addi(i,j)=A(i,j)+B(i,j);
7             subtr(i,j)=A(i,j)-B(i,j);
8         end
9     end
10    addi,subtr
11 end
12
13
14

Command window
>> %using function
>> a=randi(10,3,4);b=randi(10,3,4);
>> add_sub(a,b)

addi =

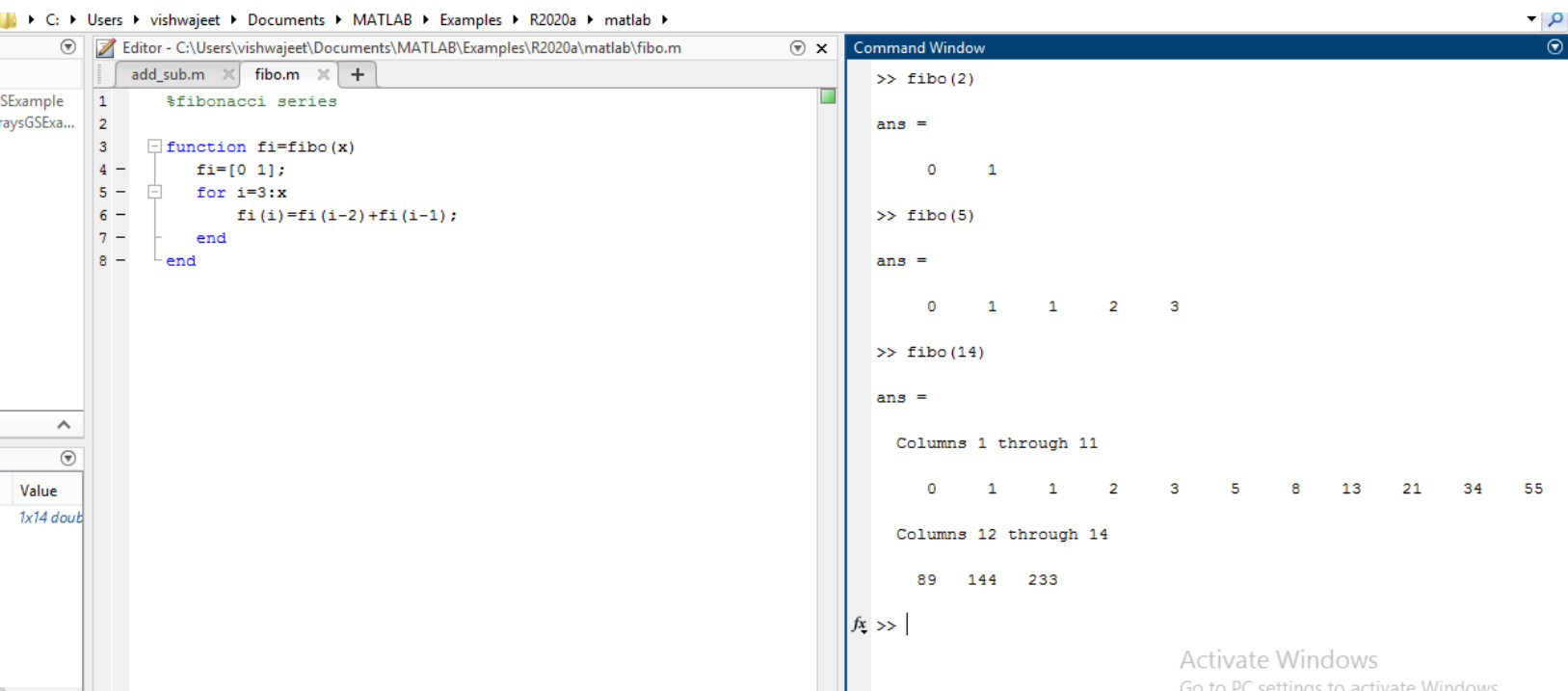
     7     12     11     17
    10     13     13     15
     7     16      8     15

subtr =

     5      6      1     -3
     4      3     -5      1
     3      4     -4     -5
```



# Q7: Fibonacci series function



The image shows a MATLAB environment with the Editor and Command Window. The Editor displays a function named `fibonacci series` in `fibonacci.m`. The function initializes `fi` as `[0 1]` and uses a `for` loop to calculate the Fibonacci sequence up to `x`. The Command Window shows the execution of the function for `fibonacci(2)`, `fibonacci(5)`, and `fibonacci(14)`, displaying the resulting Fibonacci sequence values.

```
1 %fibonacci series
2
3 function fi=fibo(x)
4     fi=[0 1];
5     for i=3:x
6         fi(i)=fi(i-2)+fi(i-1);
7     end
8 end
```

```
>> fibo(2)

ans =

     0     1

>> fibo(5)

ans =

     0     1     1     2     3

>> fibo(14)

ans =

Columns 1 through 11

     0     1     1     2     3     5     8    13    21    34    55

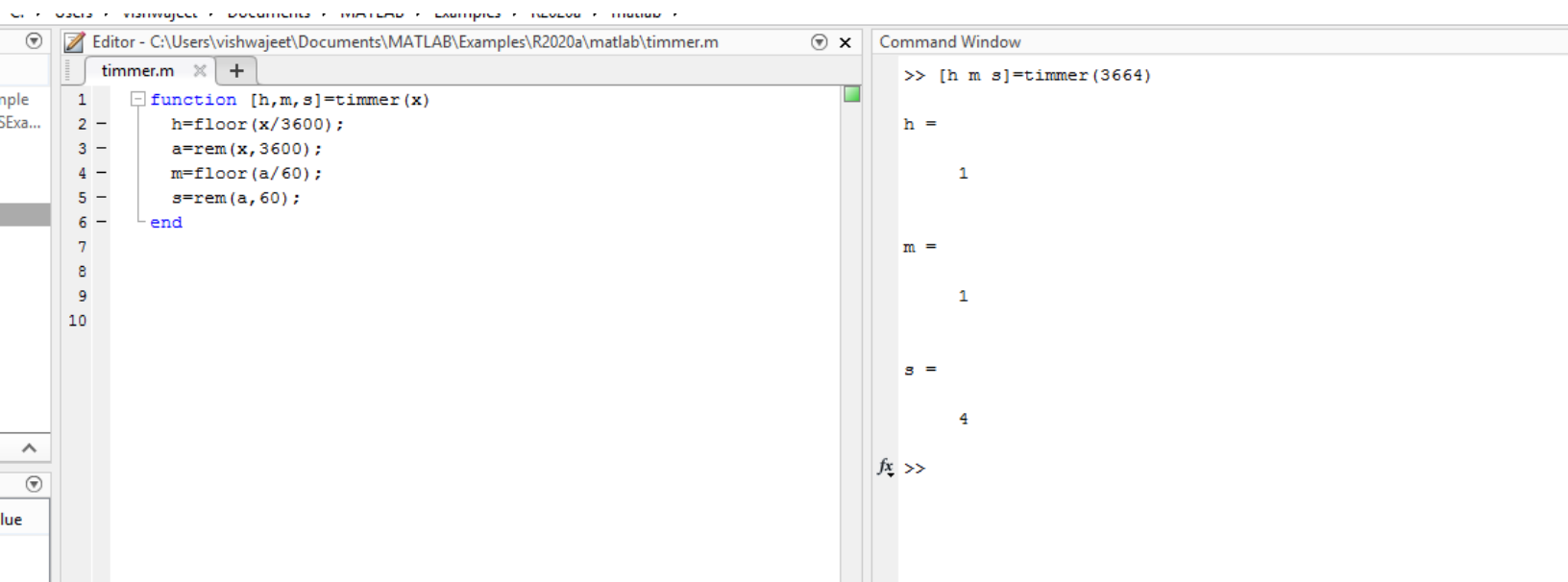
Columns 12 through 14

    89   144   233

fx >> |
```

Activate Windows  
Go to PC settings to activate Windows.

## Q8: hrs:min:sec



The image shows a MATLAB environment with two windows. The Editor window on the left displays a function named `timmer.m` (note the typo) with the following code:

```
1 function [h,m,s]=timmer(x)
2     h=floor(x/3600);
3     a=rem(x,3600);
4     m=floor(a/60);
5     s=rem(a,60);
6 end
```

The Command Window on the right shows the execution of the function with the input 3664:

```
>> [h m s]=timmer(3664)

h =

     1

m =

     1

s =

     4

fx >>
```

The output indicates that 3664 seconds is equivalent to 1 hour, 1 minute, and 4 seconds.

# Q9: mean and std of vector

C:\Users\ vishwajeet \Documents \ MATLAB \ Examples \ R2020a \ matlab \

```
Editor - C:\Users\ vishwajeet \Documents \ MATLAB \ Examples \ R2020a \ matlab \ avg_std.m
timmer.m  avg_std.m  +
1  %mean and std of vector
2  function [avg,std1]=avg_std(x)
3  -      avg=mean(x);
4  -      std1=std(x);
5  -  end
```

Command Window

```
>> x=rand(1,10000000);
>> [avg,std1]=avg_std(x)

avg =

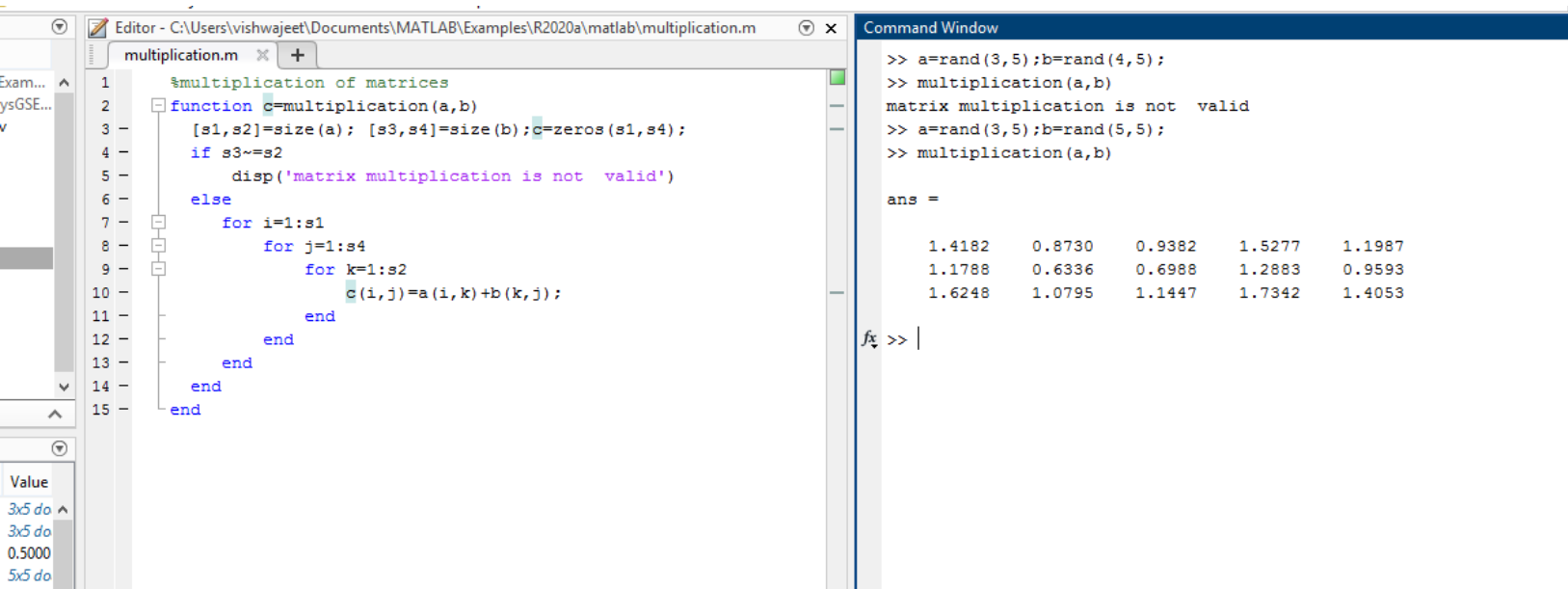
    0.5000

std1 =

    0.2887

fx >> |
```

# Q10: matrix multiplication



The image shows a MATLAB environment with an Editor window and a Command Window. The Editor window displays a function named `multiplication` that takes two matrices `a` and `b` as input. It checks if the number of columns of `a` (s2) is equal to the number of rows of `b` (s3). If not, it displays an error message. If yes, it performs matrix multiplication using nested for loops and stores the result in `c`.

```
1 %multiplication of matrices
2 function c=multiplication(a,b)
3     [s1,s2]=size(a); [s3,s4]=size(b); c=zeros(s1,s4);
4     if s3~=s2
5         disp('matrix multiplication is not valid')
6     else
7         for i=1:s1
8             for j=1:s4
9                 for k=1:s2
10                    c(i,j)=a(i,k)+b(k,j);
11                end
12            end
13        end
14    end
15 end
```

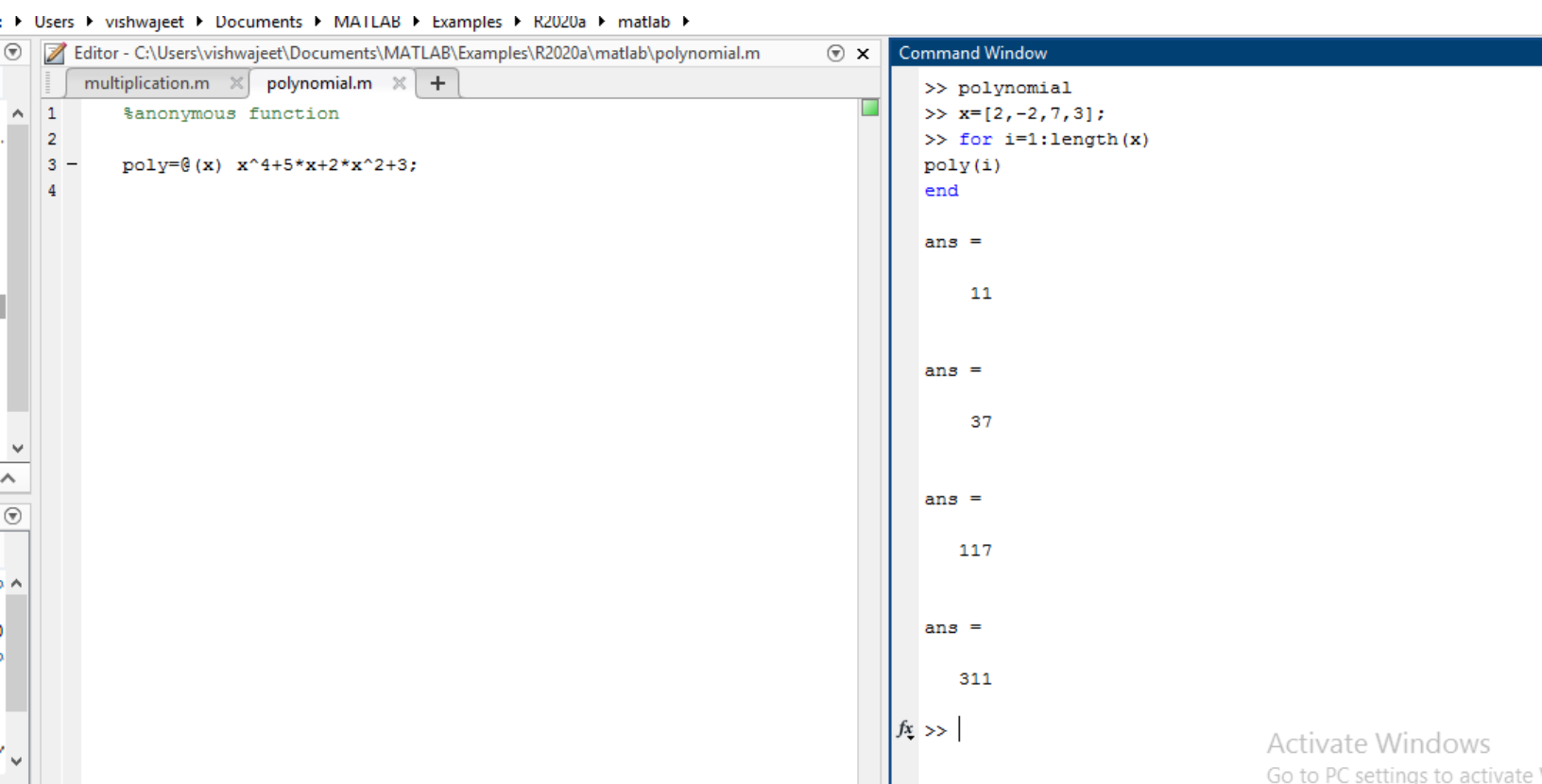
The Command Window shows the execution of the function. First, `a` is a 3x5 matrix and `b` is a 4x5 matrix. The function returns the error message 'matrix multiplication is not valid' because the number of columns of `a` (5) is not equal to the number of rows of `b` (4). Then, `a` is a 3x5 matrix and `b` is a 5x5 matrix. The function returns the result of the multiplication, which is a 3x5 matrix.

```
>> a=rand(3,5);b=rand(4,5);
>> multiplication(a,b)
matrix multiplication is not valid
>> a=rand(3,5);b=rand(5,5);
>> multiplication(a,b)

ans =

    1.4182    0.8730    0.9382    1.5277    1.1987
    1.1788    0.6336    0.6988    1.2883    0.9593
    1.6248    1.0795    1.1447    1.7342    1.4053
```

# Q11: anonymous solution of polynomial



The image shows a MATLAB environment with two windows. The Editor window on the left displays a file named 'polynomial.m' containing an anonymous function definition. The Command Window on the right shows the execution of this function for a vector of values.

```
Editor - C:\Users\vishwajeet\Documents\MATLAB\Examples\R2020a\matlab\polynomial.m
multiplication.m x polynomial.m x +
1 %anonymous function
2
3 poly=@(x) x^4+5*x+2*x^2+3;
4
```

```
Command Window
>> polynomial
>> x=[2,-2,7,3];
>> for i=1:length(x)
poly(i)
end

ans =

    11

ans =

    37

ans =

   117

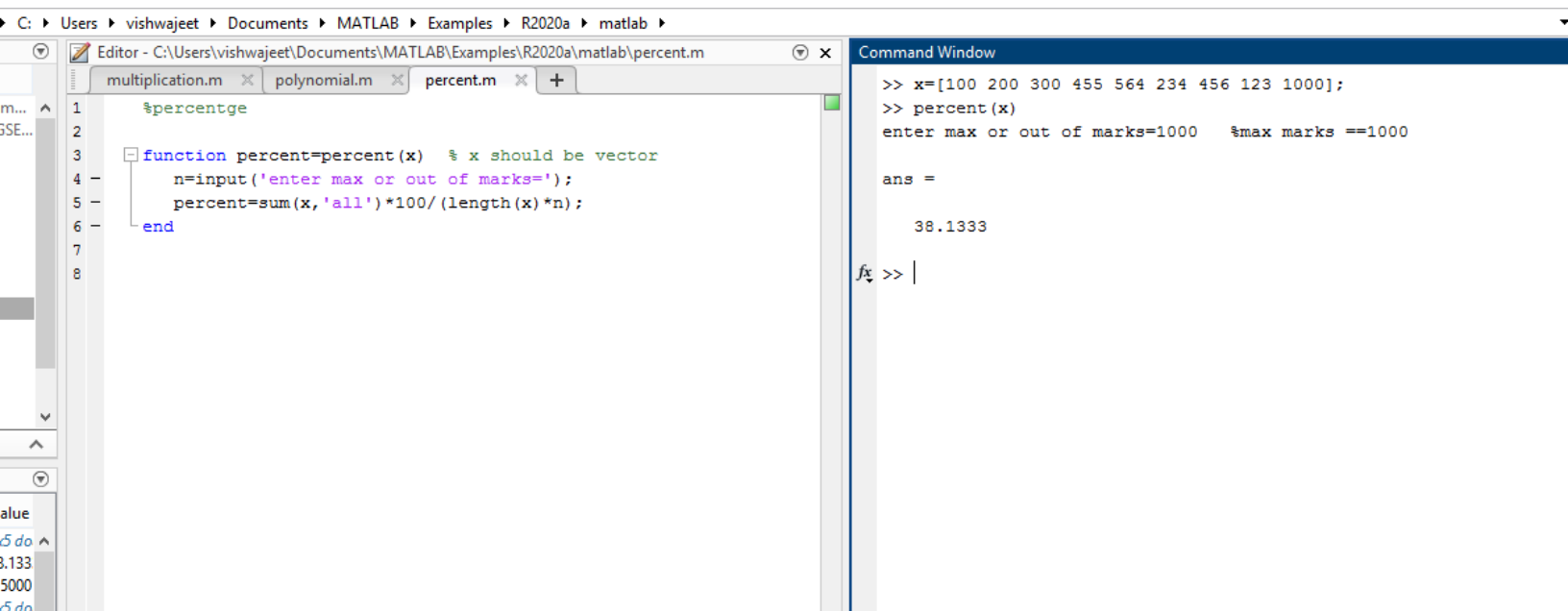
ans =

   311

fx >> |
```

Activate Windows  
Go to PC settings to activate

# Q12: percentage



The image shows a MATLAB environment with the Editor and Command Window. The Editor displays a script named `percent.m` with the following code:

```
1 %percentage
2
3 function percent=percent(x) % x should be vector
4     n=input('enter max or out of marks=');
5     percent=sum(x,'all')*100/(length(x)*n);
6 end
7
8
```

The Command Window shows the execution of the script:

```
>> x=[100 200 300 455 564 234 456 123 1000];
>> percent(x)
enter max or out of marks=1000 %max marks ==1000

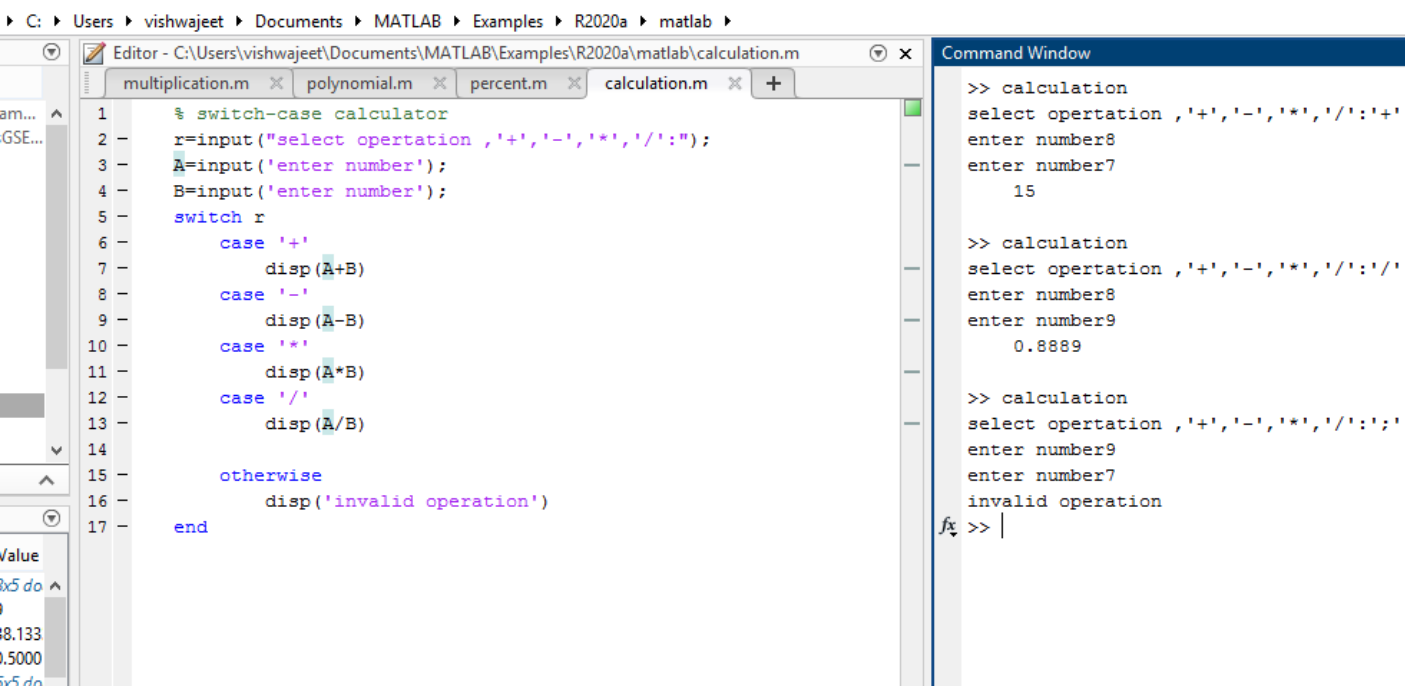
ans =

    38.1333

fx >> |
```

The output of the script is 38.1333, which represents the percentage of the sum of the elements in the vector `x` relative to the maximum possible value (1000).

# Q13: basic calculator



The image shows a MATLAB environment with the Editor and Command Window. The Editor displays a script named 'calculation.m' which implements a basic calculator using a switch-case structure. The Command Window shows the execution of the script with three test cases: addition, multiplication, and an invalid operation.

```
Editor - C:\Users\vishwajeet\Documents\MATLAB\Examples\R2020a\matlab\calculation.m
multiplication.m x polynomial.m x percent.m x calculation.m x +

1 % switch-case calculator
2 r=input("select operation ,'+','-','*','/':" );
3 A=input('enter number');
4 B=input('enter number');
5 switch r
6     case '+'
7         disp(A+B)
8     case '-'
9         disp(A-B)
10    case '*'
11        disp(A*B)
12    case '/'
13        disp(A/B)
14
15    otherwise
16        disp('invalid operation')
17 end

Command Window

>> calculation
select operation ,'+','-','*','/':"
enter number8
enter number7
      15

>> calculation
select operation ,'+','-','*','/':"
enter number8
enter number9
    0.8889

>> calculation
select operation ,'+','-','*','/':"
enter number9
enter number7
invalid operation
fx >> |
```